AFOSR-TR-95

0586

REPORT DOCUMENTATION PAGE

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE 18th February 1995	3. REPORT TYPE AND DATES COVERED FINAL

4. TITLE AND SUBTITLE
Reactive Execution in a Command, Planning and Control Environment

5. FUNDING NUMBERS
AASERT -
F49620-93-1-0436

6. AUTHOR(S)
Glen A. Reece

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Edinburgh
80 South Bridge
Edinburgh
United Kingdom, EH1 1HN

8. PERFORMING ORGANIZATION REPORT NUMBER
AASERT/O-PLAN/AR/2

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NM
110 Duncan Avenue, Suite B115
Bolling AFB
DC 20332-6448
USA

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

DTIC
S ELECTE D
OCT 1 0 1995
G

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This research proposed a design of a reactive execution agent (REA) which accepts and executes task directives in a dynamic environment. This design drew upon an amalgam of ideas emerging from situated rational agency research. The result was an agent which was able to accept tasksdirectives (or reject them), reason about directives to determine how to achieve them (maintaining commitments to achieving other directives), respond to execution failures, and communicate with a superior agent when further deliberation is required beyond the abilities of the REA.

The primary contributions of the research have been 1) a characterised model of rational behaviour 2) an inter-agent communication language (IACL) for adapting execution time behaviour, and 3) the use of causal structure to predict potential execution failures. The research has shown how reasoning about commitment to tasks, failure related to tasks, and tasks with temporal extent interact with the basic set of capabilities to provide a robust model for competent and rational situated behaviour.

DTIC QUALITY INSPECTED 5

14. SUBJECT TERMS
Reactive execution, rational behaviour, inter agent communication

15. NUMBER OF PAGES
25

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

AASERT/O-Plan/AR/2

Reactive Execution in a Command, Planning and Control Environment

Final Technical Report

Prepared by
Glen A. Reece, Ph.D.

Department of Artificial Intelligence
The University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN
United Kingdom

18 February 1995

Contract
AASERT F49620-93-1-0436

Prepared for
Dr. Abraham Waksman
AFOSR/NM
Bolling AFB

Contents

List of Figures

1 Summary

Carrying out previously developed plans in the real world, such as picking a friend up at the airport or cooking dinner, requires that we possess some basic abilities for behaving rationally. As we all know, things just never go to plan 100 percent of the time, so possessing the facilities to cope with this uncertainty makes us good execution agents. However, the term "rational" is easy to state, and difficult to define. Just what abilities do we possess that allow us to reason in a rational manner?

Researchers in the field of Artificial Intelligence (AI) have been attempting to make a precise definition of rationality since the beginning. Though we still do not have a universally accepted definition, we do have systems which to varying degrees appear to demonstrate qualities which can be said to be rational. If we characterize those systems, we can define a minimum set of capabilities which allow us to design a basic rational agent.

This research proposed a design of a reactive execution agent (REA) which accepts and executes task directives in a dynamic environment. This design drew on an amalgam of ideas emerging from situated rational agency research. The result was an agent which was able to accept task directives (or reject them), reason about directives to determine how to achieve them (maintaining commitments to achieving other directives), respond to execution failures, and communicate with a superior agent when further deliberation is required beyond the abilities of the REA.

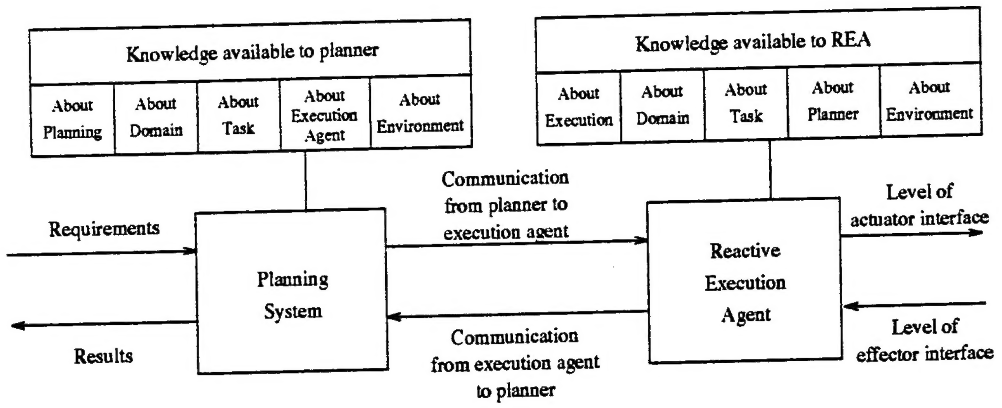
The primary contributions of the research have been (1) a characterized model of rational behavior, (2) an inter-agent communication language (IACL) for adapting execution-time behavior, and (3) the use of causal structure to predict potential execution failures. The research has shown how reasoning about commitment to tasks, failures related to tasks, and tasks with temporal extent interact with the basic set of capabilities to provide a robust model for competent and rational situated behavior.

2 Introduction

Over the past two decades researchers have shown how to design planning agents that can reason about time and resources, respond within some guaranteed time bound, respond immediately with some action or reaction, and even perform various types of re–planning in the face of unanticipated events or failures. As a result, agents are certainly better at reacting, but they have no ability to modify their capabilities, reason about the capabilities of other agents, reason about the knowledge of other agents, reason about commitment and thus, they possess no facilities for behaving intelligently.

The purpose of this research has been to address the "open" issues involved in attempting to provide agents with abilities to behave intelligently. That is, this research investigated how agents could communicate capabilities, knowledge, commitment and priority information, and reason about each of these in a dynamic, command, plan and control environment.

The focus of this research has been in dynamic domains where the demand is for a system that can take a command request, generate a plan, execute it and react to simple failures of that plan, either by repairing it or by re-planning. This was investigated in the context of two agents[1] with different roles and with possible differences of requirements for processing capacity and real-time reaction. Here, the two agents involved are a central planner and a remote Reactive Execution Agent (REA). The central planner (agent-1) has knowledge of the



Knowledge available to planner
About Planning
About Domain
About Task
About Execution Agent
About Environment
Knowledge available to REA
About Execution
About Domain
About Task
About Planner
About Environment
Requirements
Results
Planning System
Communication from planner to execution agent
Communication from execution agent to planner
Reactive Execution Agent
Level of actuator interface
Level of effector interface

Figure 1: Communication between Planner and Reactive Execution Agent

general capabilities of the remote REA (agent-2), but does not know the details of the REA's capabilities or the details of the actions required to carry out the desired task. It communicates a general plan to achieve a particular task, and responds to failures fed back from the remote REA (see Figure 1).

The REA seeks to carry out the detailed tasks specified by the planner while working with a more detailed model of the execution environment than is available to the planner. It executes

[1]A two agent scenario uas considered during this research in order to limit the issues to be explored to only those described here. However, the results are general enough to be easily scaled up to multi-agent environments.

the plan by choosing the appropriate activities to achieve the various sub-tasks within the plan, using its knowledge about the particular resources under its control. It communicates with the real world by executing the activities within the plan and responding to failures fed back from the real world. Such failures may be due to the inappropriateness of a particular activity, or because the desired effect of an activity was not achieved due to an unforeseen event. The reason for the failure dictates whether the same activity should be re–applied, replaced with other activities or whether re-planning should take place.

2.1 Objectives

The aim of this research has been to develop a system which exhibited the following characteristics.

- The ability to acquire new knowledge and capabilities. That is, be able to acquire additional knowledge from the environment via sensors, and from the planning agent through Inter-Agent Communication Language (IACL) messages. Also, be able to acquire new functionality in the environment via IACL messages from the planning agent.

- Possess a reasoning mechanism which allows the REA to reason about commitment of the planner to tasks, priority of tasks, knowledge and capabilities of itself, and about goal and fact driven entities.

- The ability to accept plans which constrain or partially constrain action selection. The REA will tend to be myopic when having to make a decision at execution time yet, it should choose the most appropriate for the immediate circumstances. The planner on the other hand has the ability to take more global constraints into consideration. Therefore, the agent should be able to constrain its selection when directed by the planner and be free to make what it feels is the best when the planner does not provide any constraints.

- Be independent and functionally separated from the planning agent yet loosely coupled to be able to receive assistance when required.

- Have the ability to recover from failures either through replanning tactics or assistance from the planning agent.

- Possess all of the REA capabilities—guaranteed response, failure recovery, innate behavior, asynchronous events, weighing alternatives, change of focus, predictability, and temporal reasoning as defined by this research.

3 Results and Discussion

The primary contributions of this research are presented in the following sections. We begin in Section 3.1 with a discussion on how rational behavior was characterized for this research and what was learned from that characterization. In Section 4 we briefly summarize the contributions of the research. Finally, in Section 4.1 we present future research directions and point out areas where this specific research might be enhanced.

3.1 Characterization of Rational Behavior

The primary issues in designing an agent that is to behave rationally in dynamic environments are choosing architectural features (e.g., interrupt handling, reactivity, etc.) required by the domain and determining how the agent's beliefs, goals, and intentions will effect its deliberation in deciding what to do next. Thus, there are important questions which need to be addressed. Questions such as, given all the possible choices of features an agent may possess, can we define a minimum set of characteristics which are desirable to achieve rational behavior?

In order to answer these questions we considered the characteristics of rationality in execution systems that have been developed over the past twelve years. The objective was to define a basic set of features which together allow a system to demonstrate behavior that is rational. We could then use these characterized features as the basis for the design of an execution system that will behave rationally.

A set of characteristics were presented that were defined to be necessary for agents to act rationally and predictably in complex and dynamic environments [Reece 1994b,Reece 1994a]. That set included requirements (or capabilities) for guaranteed response times, handling asynchronous events, weighing alternatives to manage uncertainty, a change of focus-of-attention mechanism, predictability, failure recovery, innate behavior, and a temporal reasoning facility.

As a result of the comparative evaluation conducted in this research and of trying to design an agent which incorporated each of the characteristics into a new style of architecture, it was determined that the definitions were not detailed enough to isolate the particular desired behavior. What was needed was a clear specification of the types of observable behavior that a particular characteristic may exhibit so that we could quantitatively measure the degree to which a system possesses that characteristic and indentify the best approaches. Thus, enabling us to design better systems.

Here we present the understanding we have gained through this research in developing an enhanced characterization that can be utilized in future designs of rational execution agents. We begin by extending the definitions in the original characterization to detail the particular behaviors that together compose a characteristic. We then extend the characterization itself to include other aspects of rational behavior that have been identified as significant by this research.

3.2 Enhanced Characterization

Guaranteed response was defined as the ability to guarantee some kind of response by the time a response is required. However, a time interval of zero would be truly impossible—there is *some* computation required, no matter how trivial, that must go on before an agent decides what to do. Hence, we relaxed this definition to allow for random responses, default responses, and bounded responses. Nonetheless, because doing nothing is a valid response (and very often the right response) we would also need to add that to the definition, thus yielding the undesired effect of making every architecture possess the guaranteed response characteristic. The original idea behind this characteristic was to be able to guarantee a response to an internal or external event and continue to function whether the agent was able to handle the event or not. The definition became murky when we tried to get it to double as a criterion for providing a response that was timely based upon the environment in which the agent was situated. In an attempt to clarify the definition we confine it to mean the ability of the agent to provide a response to an event in bounded time as defined according to the reasonable temporal horizon for the environment in which the agent is situated. The response can be default, random, or monotonically improving in time towards the best possible response. We make the requirement for continued operation a separate characteristic since it lends itself more to the issue of robustness than a particular type of response.

The failure recovery characteristic had the simplistic definition of the ability of the agent to continue to operate after a failure was detected either as the result of a task failure or an exogenous event. Though the continuous operation of an agent is a worthy goal, this definition has little, if anything, to do with failure recovery, and as such, has been the source of a great deal of ambiguity and confusion. What we need is a pragmatic definition of how failures could possibly be managed because we cannot in the foreseeable future, design the perfect omnipotent agent [Hallam 1994]. Hence, we concern ourselves here with whether the agent is able to recover from cognizant (or anticipated failures) and whether it can gracefully handle unanticipated failures without catastrophic collapse. When we talk about extending the characterization in Section 3.3, we will address the specific aspects of this behavior which should be exhibitable by agents that possess this characteristic.

Innate behavior was another definition that did not expressly state its intent. The original definition was that the agent be able to act without an explicit plan directing it to act. Though this definition captured the essential concept we wanted, it failed to explicitly specify the other aspects of such behavior. The intention behind this characteristic is that an agent be provided with some means to be self–sustaining in the environment in which it is situated by possessing behaviors which would allow it to function competently until such time as it receives a plan specifically directing it to act. In [Drummond and Bresina 1990] they talk of prior behavior competence (or behavioral constraints), [McDermott 1992] talks about default reactive plans, and [Lyons and Hendriks 1992] talk about an abstract plan. These systems capture the essence of the innate behavior characteristic.

The definition of the asynchronous events characteristic captures the intent of the characteristic. In the definition by [Laffey *et al.* 1988] they include the statement, "[the agent] must also be capable of processing input according to importance, even if the processing of less important input must be interrupted or rescheduled". This tends to lead one to think about the way asynchronous events will be processed and begins to obscure the definition. Here we will only be concerned with whether the agent can accept events while processing or accepting events simultaneously from different sources. We leave the issue of how the events are processed to the definition of the change of focus characteristic.

The definition of weighing alternatives fulfills its purpose as well. What we failed to do previously was to define the aspects of this behavior so we can identify this specific behavior in other systems. What we want to determine is whether an agent bases its decisions on present environmental context, heuristics about the domain, temporal deadlines, resource utilization levels, or some other set of factors.

Change of focus was defined as the ability of the agent to focus processing attention on important tasks even if the processing of less important tasks must be rescheduled or aborted. This definition implicitly states that the agent must possess the ability to determine the importance of events and that it should process the most important tasks first. To strengthen the definition, we make this latter implicit statement explicit in order to characterize the exact behavior we expect our systems to exhibit.

Predictability is the most ambiguous characteristic of our set. Do we mean predictability to imply a phenomenological model[2] where knowledge about future states can be mathematically determined? That is, action X always produces Y 97% of the time. Or do we mean that predictability implies a chain-of-events model? Where given the chain of events: action A, then B, then C we can expect response G. The intended definition is effectively a combination of these two models. What we want to be able to define is that for any input we can determine the output, not that we can predict the order of processing or interaction involved when a series of inputs are received at one time. We simply want to be able to say that we can "predict" that we know what the system will do for any particular situation taken in isolation. This is of course not enough by itself to guarantee coherent behavior when the system is actually deployed. Therefore, we shall refer to this definition as *simple* predictability, and define the term *complex* predictability for systems that are able to (1) predict the compound interaction of two or more actions, or (2) mathematically predict the occurrence of a particular action.

Finally, we consider the definition of the temporal reasoning characteristic. Again the definition satisfactorily captures the general intent while not being specific enough to encompass all of the specific intended nuances. Let's add to the definition that the agent should comprehend point and interval temporal representations, clipping, persistence, and all with respect to tasks, subtasks, and high–level plans. We want to capture the fact that the agent is completely able to reason about the aspects of time. This definition may now be too strict since temporal reasoning to this degree might not be required in all domains to achieve rational behavior. Therefore, we define this characteristic as the ability of a system to sufficiently represent temporal constraint information such that temporal relations, preferences, and constraints can be made with respect to tasks, subtasks, and high–level plans. If a particular system then chooses to include more advanced temporal reasoning mechanisms then so be it, as long as, this definition can be satisfied.

[2]This term is credited to Oliver Sparrow.

3.3 Extending the Characterization

The characterization has been extended in two areas which have received considerable attention in this research and in the AI literature as of late, and in one area we alluded to in the previous section. These are the characteristics of adaptability, failure management, and continuous operation respectively.

Considering the research presented here and that undertaken by [Drummond and Bresina 1990,McDermott 1992] and [Lyons *et al.* 1991] on incremental adaptation and behavior transformation we define a characteristic for adaptability. It is defined as the ability of the agent to dynamically modify its behavior, add new behaviors, add (or modify) domain knowledge, and add (or modify) procedural knowledge to address specific situations when directed to do so by a superior agent. This characteristic reflects the fact that as the environments in which we situate agents become more dynamic and complex we must have the ability to dynamically adapt the behavior of those systems to address new and novel situations not previously encountered or considered.

A characteristic involving failure management is a refinement of the failure recovery characteristic to achieve a pragmatic way of specifying that rational systems be able to address execution failure. The purpose is to identify some internal aspects of failure management that will allow us to design agents whose approach to failure management is pragmatic. Therefore, we desire that an agent possess the ability to recover from cognizant and unanticipated failures, actively monitor for protection interval violations (i.e., monitor the execution of the plan), repair execution failures by re–ordering, removing, or inserting tasks, and to utilize clean–up procedures when tasks fail.

In the previous section we stated that the ability of an agent to continue to operate was an important behavior, but that including it as part of the defining criteria for the guaranteed response or failure recovery characteristics was inappropriate. We shall then extend the characterization by adding a continuous operation characteristic. It is defined as the ability of the agent to continually operate until directed to cease operation or upon the occurrence of a catastrophic event. This definition is a re–statement of the continuous operation characteristic presented in the characterization of real–time knowledge–based systems by [Laffey *et al.* 1988].

We summarize the extended and final characterization of rational behavior in Table 1.

The problem with arbitrarily adding more and more characteristics is that we end up including everything as necessary for rational behavior (and blurring the distinction of sufficient for such behavior). For instance, we could include resource reasoning, qualitative reasoning, goal–directed behavior, and the ability to minimize actions not in service of higher level goals as additional characteristics (just to name a few). It does not seem unreasonable, but we must consider the problems that such additions have on the processing resources of an agent. This is a problem which is presented as an area where more research needs to be conducted.

Characteristic	Internal Behavior
Guaranteed Response	Default Response Random Response Bounded Response
Failure Management	Recover from Cognizant Failure Recover from Unanticipated Failure Monitor Protection Intervals and Execution Repair Execution Failures Locally Utilize Clean–up Procedures on Failure
Innate Behavior	Behavioral Competence Stimulus–Response Network Active/Passive Behaviors
Async. Events	Input/Output During Processing Simultaneous Input/Output
Weighting of Alternatives	Contextually Heuristically Temporally (closest to deadline) Utilization of Resources
Change of Focus	Prioritized Processing
Predictability	Stimulus–Response Correlation Task Effect Predictability Temporal Horizon Predictability Series Response Correlation
Temporal Reasoning	Point/Interval Relations (Tasks) Point/Interval Relations (Subtasks) Point/Interval Relations (Plans)
Adaptability	Add/Modify Processing Behavior Add/Modify Domain Knowledge Add/Modify Procedure Knowledge
Continuous Operation	Operate Until Directed to Halt Operate Until Catastrophic Failure

Table 1: Extended Characterization of Rational Behavior

4 Contributions

The focus of this research has been to develop a set of characteristics which describe rational behavior for complex and dynamic environments that will allow for the design of quantitatively better execution systems. This dissertation contributes towards that goal in the following ways.

Characterization of Rational Behavior. The first contribution is an explicit characterization of rational behavior. This characterization is a specification of the type of behavior that an agent architecture should provide if it is to yield an agent that behaves rationally in complex and dynamic environments over a wide variety of domains. We originally identified eight characteristics and validated that set by comparatively evaluating a representative sample of existing execution systems against that set. It included characteristics for guaranteed response, failure recovery, innate behavior, asynchronous events, weighing alternatives, change of focus, predictability, and temporal reasoning. We then extended that set to include the characteristics of adaptability and continuous operation, and replaced the failure recovery characteristic with that of failure management to establish a more pragmatic definition. The final version of the characterization specifies ten characteristics that form the basis of a guideline for the design of subsequent execution systems.

Inter–Agent Communication Language. The second contribution is a communication language that provides information to an execution system and allow for the dynamic adaptation of the behavior of an execution system. The simple message types of IACL provide a significant contribution in demonstrating how an execution system can be tasked, acquire and assimilate new or modified domain knowledge, and dynamically adapt its processing knowledge and capabilities at run–time. The definition of the language also provides for the dynamic specification of new message types that allows the language to be extended or adapted for novel domains.

Failure Management. The third contribution is a method of synthesizing protection monitors from causal structure information and a means to allow these monitors to be used to identify potential execution failures. This latter concept allows the execution system to detect potential failures early so as to provide a planning system (which must address the failure) with a greater amount of time to initiate a repair plan. Additionally, the concept of active sensing was introduced to show how this technique can be guaranteed to identify potential execution failures within specific temporal horizons.

Flexible Architecture. The fourth contribution is the validation of an agenda–based architecture as a flexible means to integrate characteristics of rationality (i.e., the O–Plan architecture). The architecture is dynamically adaptable and modular. This architecture uses knowledge sources to represent processing capabilities, and provides underlying mechanisms for asynchronous control necessary for reactive execution. The architecture is independent of the representation and thus, can be used to explore a variety of control strategies.

The Pacifica Simulator. The fifth and final contribution is a complex and dynamic environment for testing agent designs. It provides an environment for studying execution systems in the context of transportation logistics problems, and allows for remote sensing, complex object interactions, continuous time, dynamic reporting of task completion and failure, asynchronous task execution, and probabilistic task durations that are of extended length. Along with the characterization, the testbed provided by the Pacifica Simulator will allow us to quantitatively evaluate execution systems.

4.1 Future Research

Research in new areas such as intelligent execution systems tends to raise as many questions as it answers. The research in this dissertation suggests further research in two primary areas: first, in a general fashion, to develop additional reasoning capabilities for execution systems; second, to make specific additions or modifications to the proposed system to address some of its limitations or inefficiencies.

4.1.1 General Research Topics

Due to the breadth of the problem studied in this research, some of the ideas originally planned for could not be adequately addressed. Thus, several topics are left as open research and deserve separate attention. These are described in this section.

First, facilities for full temporal reasoning need to be integrated into the REA. Originally, the Time Map Manager (TMM) software [Boddy 1991,Hon 1992] developed by Honeywell was to meet this need. TMM is a nonmonotonic, deductive, temporal database management system that provides many mechanisms that would allow an agent to possess powerful reasoning mechanisms. The intent was to take advantage of TMM's rich representation of time and allow an agent to reason about time points, time intervals, persistence, clipping, and temporal projection. The theory is that with such mechanisms available an agent could reason about more of the consequences of its actions and potentially become more robust (i.e., reporting failure less frequently). Needless to say, we did not have the opportunity to explore this avenue of research. However, since no other known execution systems possess these temporal reasoning features it would be an interesting research exercise to see if in fact, the theory could be shown to be true.

Resource reasoning capabilities should be part of any intelligent execution system. An agent should be able to reason about utilization levels of resources under its control and be able to schedule task execution to avoid resource contention. The simplest approach is to use semaphores, since these do not depend upon a representation of time. However, such an approach limits the depth of reasoning that can be achieved, and it not satisfactory for all types of domains [Miller 1985]. The best approach would be to couple a resource reasoning facility with that of a full temporal reasoning facility (such as TMM) to reap the greatest benefit from both facilities. This is another interesting topic that has received little attention in the literature on intelligent execution systems. Perhaps the literature on scheduling systems would be a good place to start.

Another interesting avenue for research might be adding contingency planning capabilities to the execution system. In the case of the REA executing NEO plans in Pacifica, there were often periods where the REA was idle due to the extended duration of the tasks it was executing. We might be able to use this time to develop contingencies for currently executing tasks to better address failures should they occur. The ERE system [Drummond and Bresina 1990] uses temporal projections to find optimal solution paths, but this information can also be used to address failures. Another interesting idea along the same lines would be to explore the benefits of specifying plans which contain contingencies to the execution system.

When discussing execution systems people are usually concerned with monitoring the execution of tasks to make sure that preconditions are satisfied or that effects have been achieved. However, it might be interesting to monitor the progress of a task towards achieving its goal (e.g., effects). This idea was discussed with Mark Drummond where we talked about an agent having expectation models for the tasks it could execute or for particular tasks that had a high probability of failure. By possessing such models the agent would be able to compare its status with that of the expectation model for an executing task at prescribed intervals during execution to determine if progress was being made towards it goal. If the tasks varied some degree beyond a threshold from what was expected then the agent would then be able to intervene and hopefully, get the execution back on track. Hart at the University of Massachusetts did some work along these lines called envelopes for the Phoenix system [Hart *et al.* 1990], and Kohout at the University of Maryland also has ideas on this subject [Kohout 1993] following from the Dynamic Reaction Model of [Sanborn and Hendler 1988].

In Erann Gat's dissertation [Gat 1991], he talks about the Wesson Oil problem. In this problem, a woman is frying chicken (in Wesson Oil) when one of her children suddenly falls down and has to be taken to hospital. However, before going she turns off the stove. When she later returns she resumes frying the chicken. This is an interesting behavior which he addresses with an "unwind–protect" feature. This unwind–protect defines a clean–up procedure that should be executed when a high priority task interrupts the execution of a lower priority task. His solution, admittedly, is only a partial solution since the selection of the clean–up procedure is context dependent. However, it is a serious problem which deserves further consideration, and could potentially provide an execution system with a great deal of robustness in the face of execution failure.

The purpose of this section is to identify areas where there is potential for someone to address open research issues in the area of execution system design. The problem is that if someone were to develop each of the ideas from this section, we could easily build so much reasoning power into the execution system that we would again have the problem of trading deliberation time for reaction time. Therefore, the most interesting research topic of all (in my opinion of course) would be to identify the trade–offs so future designers could use that knowledge. For example, consider resource reasoning. How much of a benefit would it actually be? If we were to spend time deliberating about resources that would avoid problems arising from resource contention later, would that be better than addressing the resource contentions as they occurred? These issues may not be addressable in a general way and may require significant domain knowledge.

4.1.2 Extending the REA Approach

First, and foremost, to extending the REA approach is to make the necessary modifications for controlling an autonomous robot or a robotic cell. This would involve developing the domain knowledge and a new Dispatch capability to interface to the hardware. This would lend credibility to the design and architecture, and identify further limitations to be addressed.

The active sensing mechanism needs to be modified in two significant ways. First, in the way that the Sensor capability determines if a particular sensor request should be dispatched to the environment. Presently, if the sensor request is for the same resource that the sensor was last used to gather information from then it is ignored[3]. This approach does not guarantee what it should. It guarantees that no two sensor requests will be made for the same resource, but what it should guarantee is that for N resources that the sensor requests would be $R_1, R_2, \ldots, R_N$ and when R_N was reached that the next request would start again with R_1. Second, we could be smarter about the number of active behaviors that are created for active sensing. At present, given the patterns (at gt1) and (res-status gt1) from two distinct causal structure records we would have two active behaviors created that both require sensing of the gt1 ground transport resource using the same sensor. The problem being that the gt-sensor gathers information for both the *at* and *res-status* attributes so having separate sensor requests is redundant and unnecessary overhead. What we should do is check, at the time we are creating active behaviors from the causal structure information, that we are not already requesting information from a particular sensor. We would however, have to concern ourselves with the fact that by not issuing the requests separately that when we no longer required *res-status* information after, say node-4, that we could still continue to gather location information until node-23.

Another area where we might be able to provide additional insight to the REA is by rating tasks as to the seriousness of failure. For example, if the airport was under attack when a plane was supposed to take–off, then the fact that we did not get clearance from the tower does not mean that the fly–transport task should fail. Also, the failure of a sensory task should not cause an entire plan to fail to execute, as is presently the case. We need some method of representing such information to the execution system so that it can make decisions regarding the severity of an execution failure. This would help to make the system more robust and reduce the frequency of the execution system having to request assistance.

A mechanism to allow the execution system to reason about the effects of the task specifications in its domain knowledge would also help to make the system more robust in the face of failure when precondition failures occur. In the current system, if a ground transport resource is to drive from one location to another and its mechanical status is bad, then the drive task will fail because all of its preconditions have not been satisfied. This causes the REA to send an IACL execution–failure message to the planning agent. If the REA could reason about the effects of other tasks in its domain knowledge (which is already included in the TBL representation of tasks) it would find that if it executed the task fix-gt-mech then it could locally repair the problem and continue with normal execution. Intuitively, a mechanism such as this would be a good idea, but it could potentially cause problems later if the fix-gt-mech task required the use of a limited resource that, unbeknownst to the REA at the time, was required later in the plan. This highlights the problem with myopic decisions taken at execution time versus allowing the planning agent to consider global concerns. [Drummond and Levinson 1992] have begun to look at how a planning agent can monotonically increase the effective performance of an execution system. What is not clear is whether the execution system can increase the effective performance of a planning agent by making local decisions. Perhaps there is a class of decisions that the execution system could make locally that would not be a detriment to some

[3]This is true except when there is only one resource of a particular type that requires sensing. For example, if information is to be gathered from only one air cargo transport then each time the Sensor capability gets a sensor request for it, the request is made.

other concern later in the plan.

A limitation in regards to the communication abilities of the REA concerns the interaction of the Guard in the Communication Manager with message events. We discussed in this research how the Guard analyzes the contents of messages to determine whether the REA will understand the information contained in the message and have the capabilities to process the message. The problem is that the design of the Guard only allows it to analyze the information contained in an IACL Synthesize message. There should be a means to dynamically specify that the Guard analyze other types of messages if need be. This is not a major research issue; however, if the REA is to be truly adaptable it should not have any limitations such as this one[4].

Lastly, another important area of future research involves how execution failures are managed by the REA. In this research we discussed some of the ways that the failure management capabilities of the REA might be improved. We will not shed any more light on the subject here except to again emphasize the importance of such research. This is another area where significant gains in building execution systems could be made.

[4]In the general O–Plan architecture it is intended that this limitation be eliminated by allowing the Guard to be "programmed" by using the details of the installed knowledge sources given by a knowledge source formalism which has yet to be fully specified [Tate 1994].

5 Status and Conclusions

This research began with the desire to address some of the open research issues related to execution system design. It soon became apparent however, that there was no single example of a system that could be used as *the* model for a design since different execution systems offered different advantages. Thus, the efforts transformed from designing a new system to that of contributing to the way in which such systems could be better designed. In examining the AI literature to learn how to design execution systems, patterns began to emerge. Similar characteristics were identified which existed, in various guises, across implementations. This fact lead to the development of the characterization of rational behavior and the design methodology presented in this research. The work on the REA has been completely implemented and tested.

The value of the characterization presented in this research comes from the fact, that for the first time, we have a basis upon which to comparatively evaluate one execution system against another. However, the characterization falls short in several areas. First, it does not allow us to determine which characteristics are more important than others. Second, it does not provide insight into the possible conflicts which might arise when combining these characteristics in a particular architecture. What it is hoped can be taken away from this research is that the characterization does indeed define the minimum set of behaviors necessary to behave rationally. Additionally, that the ways in which these characteristics were implemented in the REA design does demonstrate that the specific design approach is one worth duplicating. What we need now is a way to determine the best implementation of a particular characteristic so we can design better architectures for rational, competent behavior in dynamic environments.

But have we truly characterized rationality? Probably not, since it could not be said that a system was irrational if it did not possess all of the identified characteristics. Perhaps a better question would be to ask whether a system should or needs to behave rationally in complex and dynamic environments to be successful in accomplishing its tasks. These questions are left open to discussion. What has been learned as a result of this research is that you must have failures in order to have progress. Maybe this attempt at quantifying rationality is incomplete, but if that allows someone else to ponder these issues and develop a better way to design and compare execution systems then this work been successful. It is clear that a great deal of work remains to be done.

6 Publications

[**Reece 1994a**] G. A. Reece. Characterization and Design of Rational Competent Execution Agents for use in Dynamic Environments. Ph.D. Thesis, University of Edinburgh, November, 1994 (257 pages).

[**Reece 1994c**] G. A. Reece. Reactive Execution in a Command, Planning and Control Environment: Annual Technical Report. Technical Report AASERT/O-Plan/AR/1, Artificial Intelligence Applications Institute, July 1994.

[**Reece and Tate 1994**] G. A. Reece and A. Tate. Synthesizing Protection Monitors from Causal Structure. In *Proceedings of the Artificial Intelligence Planning Systems Conference*, Chicago, IL, 1994.

[**Reece 1994**] G. A. Reece. The Pacifica Simulator. Technical Report ARPA-RL/O-Plan2/TR/14, Artificial Intelligence Applications Institute, February 1994. Presented at the ARPA/RL PRECiS Domain Development Group Meeting, Tucson, AZ, February 25, 1994.

[**Reece** *et al.* **1993**] G. A. Reece, A. Tate, D. Brown, and M. Hoffman. The PRECiS Environment. In *Proceedings of the National Conference on Artificial Intelligence* (AAAI-93) DARPA-RL *Planning Initiative Workshop*, Washington, D.C., 1993.

[**Reece 1993**] G. A. Reece. Rational Agency Characterization for Reactive Execution Agent Design. Technical Report ARPA-RL/O-Plan2/TP/2, Artificial Intelligence Applications Institute, June 1993.

[**Reece and Tate 1993**] G. A. Reece and A. Tate. The Pacifica NEO Scenario. Technical Report ARPA-RL/O-Plan2/TR/3, Artificial Intelligence Applications Institute, March 1993.

[**Reece 1992**] G. A. Reece. Reactive Execution in a Command, Planning, and Control Environment. Technical Report 121, Department of Artificial Intelligence, University of Edinburgh, Scotland, 1992.

7 Associated Personnel

The research was conducted by Glen A. Reece. It was associated with the Planning and Scheduling Research Group at the Artificial Intelligence Applications Institute (AIAI) at the University of Edinburgh which is funded in part by grant (F49620-92-C-0042) from the ARPA/Rome Laboratory Planning Initiative (ARPI) and EOARD (EOARD-92-0001) administered through AFOSR via EOARD.

The supervision team reflects the interdisciplinary nature of the research and involves Professor Austin Tate and Dr. Brian Drabble at AIAI, Dr. John Hallam at the Department of Artificial Intelligence, and Professor Jeff Collins at the Edinburgh Parallel Computing Centre.

Glen A. Reece

In October, 1991, Glen A. Reece joined the University of Edinburgh as a Ph.D. student in the Department of Artificial Intelligence. A U.S. citizen, Glen obtained his first degree, BSc in Mathematics with emphasis in Computer Science from the University of Texas at Arlington (1988). From 1987 to 1989 he was a Research Scientist in the Artificial Intelligence for Manufacturing Laboratory of the Automation & Robotics Research Institute. There he worked on various expert system projects for process control, uncertainty management, computer-aided instruction, and intelligent diagnostics. Upon receiving a Honeywell Industrial Fellowship in 1989, he attended Arizona State University receiving his MSc in Computer Science (1991). His research, conducted at Intel Corporation with Dr. Karl G. Kempf, involved applying situation assessment techniques to develop global strategic knowledge in order to limit re-scheduling of semiconductor production schedules. His interests include automated reasoning systems, knowledge representation, distributed systems, and cooperative decision making.

Austin Tate (Principal Supervisor)

Austin Tate is the Technical Director of the Artificial Intelligence Applications Institute (AIAI) and a Professorial Fellow at the University of Edinburgh.

Born in West Yorkshire, Prof. Tate was educated at the King's School, Pontefract and at the University of Lancaster, obtaining a First Class Honours degree in Computing Studies; and at the University of Edinburgh, obtaining his PhD in Machine Intelligence in 1975.

He currently holds an ARPA/Rome Laboratory Planning Initiative contract for work on logistics planning and is a member of the Technical Review Panel for the ARPA/Rome Laboratory Knowledge-based Planning and Scheduling Initiative. He is a member of the advisory panel for the informatics work being conducted by the European Space Agency.

Brian Drabble

Dr. Brian Drabble is a research scientist at AIAI. His research interests involve execution monitoring and error analysis; planning involving processes, and qualitative reasoning. He is a co-investigator for the parent grant. His other work includes research into Time Logics for planning and has co-authored and presented AIAI's planning and scheduling course.

John Hallam

Dr. John Hallam graduated with First Class Honours in Mathematics from the University of Oxford in 1979 and subsequently, in 1984, with a PhD in Artificial Intelligence from the University of Edinburgh. He has directed three research grants investigating architectures of and navigational techniques for autonomous mobile robots, has supervised four doctoral students to completion, one of whom studied coordination and scheduling of factory AGVs.

Jeff Collins

Professor Jeff Collins is Executive Chairman of the Edinburgh Parallel Computing Centre and Technology Advisor to the Lothian Region Development Authority in Scotland. Prof. Collins' has a distinguished career in Electrical Engineering and in the mid 1980s was Head of the Department of Electrical Engineering at the University of Edinburgh. At that time he was a member of the Advisory Board of AIAI. From 1986 to 1990 he was at the University of Texas at Arlington where he established the Automation and Robotics Research Institute. He returned to Edinburgh in 1990 to his current posts.

8 Interactions (Coupling Activities)

Presentations of this work where given to the AI Center at MITRE on November 1, 1994 and ISX Corporation on December 22, 1994.

A paper (entitled: Actively Montioring for Potential Execution Failures) has been submitted to the International Joint Conference on Artificial Intelligence further describing the active sensing and potential failure detection mechanisms.

The Pacifica Simulator has been repackaged as a stand–alone simulation package called The World Simulator (WorldSim). This package allows other researchers to test reactive execution systems in a simulated complex and dynamic environment. WorldSim will be made available via FTP on the World Wide Web on March 1, 1995.

Glen A. Reece received approval of his Ph.D. Thesis (entitled: Characterization and Design of Rational Competent Execution Agents for use in Dynamic Environments) on November 24, 1994, thus completing all of the requirements for his doctorate degree. He will offically receive his degree on July 12, 1995. Glen has taken a position at ISX Corporation as a Senior Systems Engineer where he continues to be involved with the ARPA/Rome Laboratory Planning Initiative.

References

[Boddy 1991] M. Boddy. Temporal Reasoning for Planning and Scheduling. *SIGART Bulletin*, 4(3), 1991.

[Drummond and Bresina 1990] M. Drummond and J. Bresina. Anytime Synthetic Projection: Maximizing the Probability of Goal Satisfaction. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-90)*, Boston, MA, August 1990. Morgan Kaufmann.

[Drummond and Levinson 1992] M. E. Drummond and R. Levinson. How Planning Can Help a Reactive System. Draft NASA Technical report as of January 1992, 1992.

[Gat 1991] E. Gat. *Reliable Goal-Directed Reactive Control of Autonomous Mobile Robots.* PhD thesis, Virginia Polytechnic Institue and State University, 1991.

[Hallam 1994] J. Hallam, August 1994. Private Communication.

[Hart *et al.* 1990] D. Hart, S. Anderson, and P. Cohen. Envelopes as a Vehicle for Improving the Efficiency of Plan Execution. In *Proceedings of DARPA Workshop on Innovative Approaches to Planning Scheduling and Control*, pages 71–76, San Diego, California, 1990. Morgan Kaufmann.

[Hon 1992] Honeywell Systems & Research Center, 3660 Technology Dr., Minneapolis, MN 55418. *β-TMM Manual*, 1992. Honeywell SRC Technical Report CS-R92-012.

[Kohout 1993] R. Kohout. Representing Reactive Competences for use in Hard Real–Time Systems (Ph.D. Dissertation Proposal). Technical report, University of Maryland, 1993.

[Laffey *et al.* 1988] T. Laffey, P. Cox, J. Schmidt, S. Kao, and J. Read. Real-time Knowledge-based Systems. *AI Magazine*, 9(1):27–45, 1988.

[Lyons and Hendriks 1992] D. M. Lyons and A. J. Hendriks. A Practical Approach to Integrating Reaction and Deliberation. In *Proceedings of First International Conference on Artificial Intelligence Planning Systems*, 1992.

[Lyons *et al.* 1991] D. M. Lyons, A. J. Hendriks, and S. Mehta. Achieving Robustness by Casting Planning as Adaptation of a Reactive System. In *Proceedings of IEEE International Conference on Robotics & Automation*, 1991.

[McDermott 1992] D. McDermott. Transformational Planning of Reactive Behavior. Technical Report YALEU/CSD/RR941, Yale University, 1992.

[Miller 1985] D. P. Miller. Planning by Search Through Simulations. Technical Report YALEU/CSD/RR423, Yale University, 1985.

[Reece 1994a] G. A. Reece. *Characterization and Design of Rational Competent Execution Agents for use in Dynamic Environments.* PhD thesis, University of Edinburgh, Scotland, 1994.

[Reece 1994b] G. A. Reece. Reactive Execution in a Command, Planning and Control Environment. Technical Report ARPA-RL/O-Plan/AR/1, Artificial Intelligence Applications Institute, 1994.

[Sanborn and Hendler 1988] J.C. Sanborn and J.A. Hendler. A Model of Reaction for Planning in Dynamic Environments. *Artificial Intelligence in Engineering*, 3(2):95–102, 1988.

[Tate 1994] A. Tate. The Emergence of "Standard" Planning and Scheduling System Components—Open Planning and Scheduling Architectures. In C. Bäckström and E. Sandwell, editors, *Current Trends in AI Planning*, pages 14–32. IOS Press, 1994. Proceedings of the European Workshop on Planning.